

REMARKS

Claims 1-5 are pending in the present application. By this Response, claims 1, 4 and 5 are amended for clarification purpose. Claim 2 is amended for clarification purpose. No new matter is added as a result of the amendment to claims. Reconsideration of the claims in view of the above addition to claims and the following remarks is respectfully requested.

Applicant has also submitted a formal set of drawings incorporating corrections to drawings labeled Figure 1-3 as suggested by the Examiner. Figure 1-3 are herein labeled as "Prior Art" as requested by the Examiner.

I. 35 U.S.C. § 102(e), Alleged Anticipation, Claims 1, 4 and 5

The Office Action rejects claims 1, 4 and 5 under 35 U.S.C. § 102(e) as being allegedly anticipated by Cohen et al. (U.S. Patent No. 6,434,618 B1). This rejection is respectfully traversed.

As to claims 1, 4, and 5, the Office Action states:

Referring to claim 1 Cohen has taught an method for managing communications between requesters (Col 4 line 66 – Col 5 line 6, see figure 4, item 420 is the requester) and server processes (Col 2 lines 22-34, and figure 1-4, the network element could be placed between a requester and a server.) in a data processing network including:

creating a set of dispatcher processes, each having a unique process identifier (Col 8 lines 49-50, Col 2 lines 45-54, a unique ID is assigned to packet header by the dispatch, Col 4 lines 46-48, there is more than one gateway programs, and each associated with a dispatcher, therefore a set of dispatcher processes are created);

associating each a set of requester processes, which communicate with a server process via a common interpreter process (see figure 4, the network interface which communicate between the requester and the server is viewed as a common interpreter process,) having a single process identifier (Col 8 lines 31-35, mark is viewed as a single process identifier), with a different dispatcher process of said set of dispatcher processes;

for requests sent from any of said set of requester process via said common interpreter process to server process which identifies requester processes using a process identifier, routing said requests via the associated dispatcher process (Col 7 lines 11-38);

at the respective dispatcher process, attaching the unique identifier of the dispatcher process to the request and then forwarding the request to the server process (Col 8 lines 49-56);

responsive to receipt by the dispatcher process of a reply to said request, forwarding the reply to the associated requester process via the common interpreter process (Col 4 lines 33-38).

Referring to claims 4 and 5 claims 4 and 5 encompass the same scope of the invention as that of the claim 1. Therefore, claims 4 and 5 are rejected for the same reason as the claim 1.
Office Action dated September 24, 2003, pages 2-4.

Amended independent claim 1, which is representative of independent claims 4 and 5 with regard to similarly recited features, now recites:

1. A method for managing communications between requester processes and server processes in a data processing network, including:
creating a set of dispatcher processes, each have a unique dispatcher process identifier;
associating each of a set of requester processes, which communicate with a server process via a common interpreter process having a common process identifier, with a unique dispatcher process of said set of dispatcher processes;
for requests sent from any of said set of requester processes via said common interpreter process to a server process which identifies each of said set of requester processes using the unique dispatcher process identifier, routing said requests via the associated dispatcher process;
at the respective dispatcher process, attaching the unique dispatcher process identifier to the request and then forwarding the request to the server process; and
responsive to receipt by the dispatcher process of a reply to said request, forwarding the reply to the associated requested process via the common interpreter process.
(emphasis added)

A prior art reference anticipates the claimed invention under 35 U.S.C. § 102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 21 U.S.P.Q.2d 1031, 1034 (Fed Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. *Kalman v. Kimberly-Clark Corp.*, 713 F.2d 760, 218 U.S.P.Q. 781 (Fed. Cir. 1983). Applicant respectfully submits that Cohen does not teach every element of the claimed invention arranged as they are

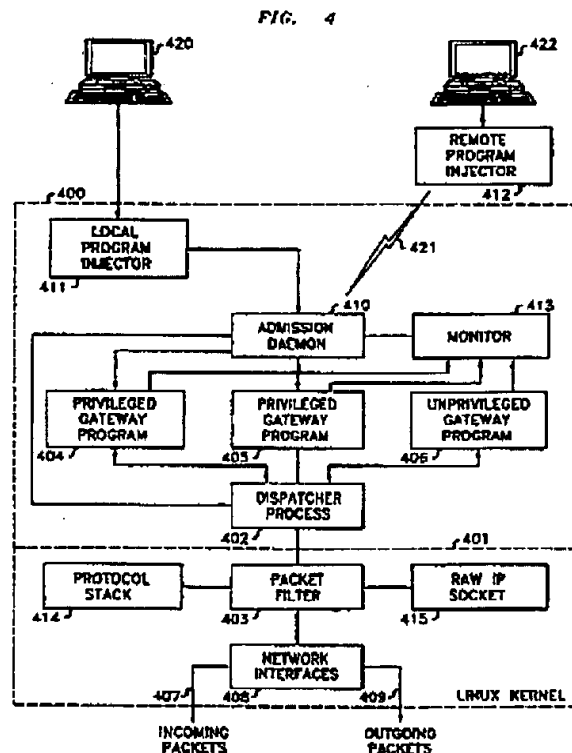
in claim 1. Specifically, Cohen does not teach creating a set of dispatcher processes, each have a unique dispatcher process identifier; associating each of a set of requester processes, which communicate with a server process via a common interpreter process having a common process identifier, with a unique dispatcher process of said set of dispatcher processes; for requests sent from any of said set of requester processes via said common interpreter process to a server process which identifies each of said set of requester processes using the unique dispatcher process identifier, routing said requests via the associated dispatcher process; attaching the unique dispatcher process identifier to the request; and forwarding the reply to the associated requested process via the common interpreter process as recited in claim 1.

Cohen teaches a programmable network element that operates on packet traffic flowing through the element in accordance with a gateway program, which is dynamically uploaded into the network element or unloaded from it via a mechanism separate from the actual packet traffic as the element operates. Such programmable network element can simultaneously operate on plural packet flows with different or same programs being applied to each flow. A dispatcher provides a packet filter with a set of rules provided by one or more of the dynamically loaded and invoked programs. These rules define, for each program, the characteristics of those packets flowing through the network element that are to be operated upon in some manner. A packet that flows from the network through the filter and satisfies one or more of such rules is sent by the packet filter to the dispatcher. The dispatcher, in accordance with one of the programs, either sends the packet to the program for manipulation by the program itself, or manipulates the packet itself in a manner instructed by the program. The processed packet is sent back through the filter to the network for routing to its destination.

However, Cohen does not teach creating a set of dispatcher processes, each have a unique dispatcher process identifier as recited in independent claim 1. The Office Action alleges that Cohen teaches a set of dispatcher processes, at column 4, lines 46-48, which reads as follows:

Under certain conditions, the monitor process 413 may request the admission daemon 410 to terminate one or more gateway programs. (Column 4, lines 46-48, Cohen)

The Office Action states that since there are more than one gateway programs, and each associated with a dispatcher, therefore a set of dispatcher processes are created. However, Cohen teaches a single dispatcher process (element 402) in Figure 4, which is reproduced below:



As shown in Figure 4, only one dispatcher process 402 is present in Cohen's system. At column 4, lines 14-19, which describes Figure 4, Cohen teaches that Dispatcher process 402 is the only process that uses the packet filter process 403 to obtain packets requested by any of the gateway programs. Dispatcher process 402 is responsible for sending incoming packets on the input of the network interfaces 408 to the particular gateway program that wish to process them, if any, and for sending the processed packets back to the kernel. Therefore, Cohen only teaches one dispatcher process, not a set of dispatcher

processes as recited in claim 1. Gateway programs 404-406 are requester processes, which send or receive packets based on requests from local client 420 or remote client 412. The gateway programs are not dispatcher processes as alleged in the Office Action.

In addition, Cohen does not teach the feature of each have a unique dispatcher process identifier as recited in claim 1. The Office Action alleges that Cohen teaches a unique id, at column 8, lines 49-56, which reads as follows:

id is a unique identifier assigned to the packet header by the dispatcher. When a gateway program sends a packet header back to the dispatcher, its identifier must remain the original identifier assigned to it by the dispatcher. The dispatcher needs this identifier in order to be able to associate the packet header with the packet payload which is buffered by the dispatcher.

(Column 8, lines 48-56, Cohen)

However, Cohen does not teach a unique dispatcher process identifier. Since Cohen only teaches one dispatcher process, as illustrated in Figure 4, no unique identifier is necessary to identify only one dispatcher process. In addition, in the above section, the id field in the structure of the header message type is used for messages carrying packet headers from the dispatcher to the gateway programs and vice versa. Thus, Cohen only teaches the id field as a unique identifier used by the dispatcher to assign a packet header, so that when the packet header is sent back and forth between the gateway programs and the dispatcher, the dispatcher can use the id to associate the corresponding packet payload. This id field is not a unique dispatcher process identifier that identifies a unique dispatcher process. Therefore, Cohen does not teach a unique dispatcher process identifier as recited in claim 1.

In addition, Cohen does not teach a common interpreter process having a common process identifier. The Office Action alleges that Cohen teaches a common process identifier at column 8, lines 31-35, which is reproduced below:

mark contains the mark associated with the flow to which this packet belongs. This mark is declared by the gateway program when it requests a packet flow from the dispatcher.

However, the mark in the above section is not the same as a common process identifier. Cohen teaches at column 8, lines 7-12, that the mark:

lets the gateway program assign marks to packet flows so that it can distinguish between packets belonging to different flows just by looking at the marks included with the packets it receives from the dispatcher process.

Thus, the mark is a field that allows gateway programs to identify the flow of a packet, whether from gateway program to dispatcher or from the dispatcher to gateway programs. The mark of Cohen is not a common process identifier that identifiers a common interpreter process, for example, a single JVM that is shared among multiple request processes. Thus, Cohen does not teach a common process identifier as recited in claim 1.

Furthermore, Cohen does not teach that requests sent from any of a set of requester processes via a common interpreter process to a server process which identifies each of the set of requester processes using the unique dispatcher process identifier, are routed via an associated dispatcher process. The Office Action alleges that Cohen teaches such features at column 7, lines 11-38, which merely points to a template data structure used for the purpose of passing information between the dispatcher and the gateway programs. The data structure does not constitute requests that are sent from a requester process to a server process as described in claim 1. As described above, Cohen fails to teach or suggest a unique dispatcher process identifier, therefore, Cohen also fails to teach or suggest a server that identifies each of the requester process using the unique dispatcher process identifier.

Moreover, Cohen does not teach attaching the unique dispatcher process identifier to the request as recited in claim 1. The Office Action alleges that Cohen teaches this feature at column 8, lines 49-56, which is reproduced above. However, Cohen only teaches the id field is used by the dispatcher to associate a packet header to packet payload (message body), as opposed to identifying a unique dispatcher process associated with the requester process, as recited in claim 1. The packet header to body association has nothing to do with the

requester and dispatcher process association. In addition, the id field of Cohen is used by the dispatcher, as opposed to by the server process to identify the requester process.

In addition to the above, Cohen does not teach forwarding the reply to the associated requested process via the common interpreter process. The Office Action alleges that Cohen teaches these features at column 4, lines 33-38, which read as follows:

The manipulated packet is then passed back through the packet filter 403 to the network interfaces 408 onto output 409 for routing to the destination address indicated in its header, which address may have been manipulated in accordance with one of the gateway programs.
(Column 4, lines 33-38, Cohen)

In the above section, while Cohen teaches a packet is passed back through the packet filter to the network interface to route to the destination in the header once the gateway programs manipulate the packet, Cohen does not teach how the response (reply) is forwarded. Cohen does not teach or suggest anything on how the response is forwarded back to the gateway programs. Thus, Cohen does not teach forwarding the reply to the associated requested process via a common interpreter process, as recited in claim 1.

In view of the above, Applicant respectfully submits that Cohen does not teach each and every feature of independent claims 1 as is required under 35 U.S.C. § 102(e). The other independent claims 4 and 5 recite similar features that are also not taught by Cohen. Accordingly, Applicant respectfully requests withdrawal of the rejection of claims 1, 4 and 5 under 35 U.S.C. § 102(e).

II. 35 U.S.C. § 103(a), Alleged Obviousness, Claim 2-3

The Office Action rejects claims 2 and 3 under 35 U.S.C. § 103(a) as being allegedly unpatentable over Cohen et al. (U.S. Patent No. 6,434,618 B1) in view of Bayeh (U.S. Patent No. 6,223,202 B1). This rejection is respectfully traversed.

As to claims 2-3, the Office Action states:

Referring to claim 2 -3, recites the limitation of a using Java Virtual Machine including respective Servlet running on a Web server, and Cohen has not taught such limitation using Java Virtual Machine.

However, all the claimed elements, such as JVM, Web server, Web application server, Servlet threads, Web browser are all well known network communication elements in the network communication arts. Bayeh shows a "A Web server that implements a Java virtual machine can be functionally extended using Java "servlets." (Col 2 lines 27-35), and also teaches dispatchers which communicates with web servers under the virtual machine environment (Col 7 lines 41-59).

It would have been obvious to a person with ordinary skill in the art at the time the invention was made to modify the teaching of Cohen such that to use a Java Virtual Machine including respective Servlet running on a Web server in his invention because Bayeh has taught a dispatcher which communicates with web servers under the virtual machine environment (Col 7 lines 41-59).

A person with ordinary skill in the art would have been motivated to make the modification to Cohen because having JVM running in Cohen's system would allow the Java programming language to be used in Cohen's invention, and Java is gaining a wide acceptance for writing Web applications, as it is a robust portable object-oriented language defined specifically for the web environment (Col 2 lines 1-26).

Office Action dated September 24, 2003, pages 4-5.

Amended dependent claim 2 reads as follows:

2. A method according to claim 1, wherein the common interpreter process via which each of said set of requester processes associated with the unique dispatcher process communicates is a Java Virtual Machine.

The Cohen reference has been discussed at length above. Bayeh teaches a system for enabling multiple virtual machine to execute on a single server, using virtual machine pooling. The integrity of an application's data will be protected from inadvertent overwriting by another application, because each application can be running in a separate virtual machine. Garbage collection, crashes, and hangs will no longer temporarily or completely halt a server: when one virtual machine halts, other can continue executing. Multiple environments can now execute on a single server, including different versions of virtual machines, increasing the mix of servlets that can be supported. Further, debugging can now occur concurrently with normal application execution, by isolating the debugging function to a specific virtual machine.

While Applicant agrees with the Examiner that Cohen does not teach a Java Virtual Machine, Applicant disagrees with the Examiner that Bayeh teaches that a

common interpreter process via which each of a set of requester processes associated with a unique dispatcher process communicates is a Java Virtual Machine. The Office Action alleges that Bayeh teaches these feature at column 2, lines 27-35 and at column 7, lines 41-49, which read as follows:

A Web server that implements a Java Virtual Machine can be functionally extended using Java "servlets". A servlet is a relatively small executable code object that can be dynamically plugged in, or added, to the code running on the server. Servlets typically perform some specialized function, which can be invoked by the server (or by another servlet) to extend its own functionality. The servlet processes the request, and returns the response to the server (or servlet) that invoked it.

(Column 2, lines 27-35, Bayeh)

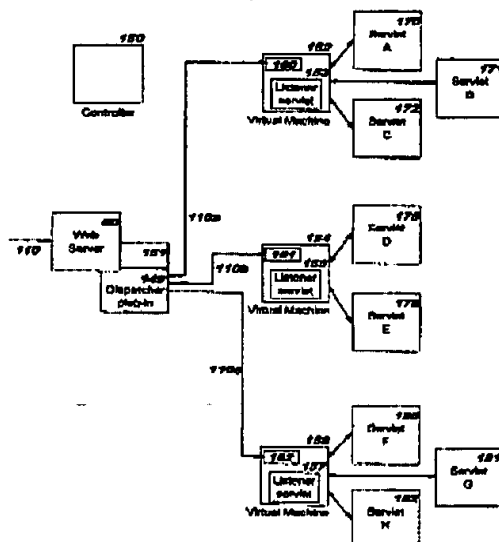
As shown in FIG. 4, the server 60 has a plug-in 151 running on it. (A plug-in is executable code that extends the functionality of the Web server, and is merely one form in which this component of the preferred embodiment may be packaged.) This plug-in 151 includes a dispatcher component 149, which receives client requests 110 from the Web server 60, and then route those requests to one of the virtual machines 152, 154 and 156. The requests are received into an application queue, where they remain until assigned to a servlet for execution.

(Column 7, lines 41-49, Bayeh)

However, in the above sections, Bayeh merely teaches a Java Virtual Machine, the functionality of which may be extended using servlets and plug-in applications. Bayeh does not teach the JVM as a common interpreter process that facilitates communications for multiple requester processes, each of which is associated with a unique dispatcher process as recited in claim 2. The JVM recited in claim 2 allows each of the requester processes to run on the same JVM in order to communicate with the server process. None of the above sections of Bayeh teaches this feature.

In addition, Bayeh does not teach that each of the set of requester processes is associated with the unique dispatcher process, as recited in amended claim 2. To the contrary, Bayeh, similar to Coheli, teaches only one dispatcher component 149 as shown in Figure 4, which is reproduced below:

FIG. 4



As shown in Figure 4, only one dispatcher plug-in 149, is used to route requests to one of the application queue in virtual machines 152, 154, 156. Thus, each of the requester processes in Bayeh shares the same dispatcher plug-in 149, as opposed to a unique dispatcher process associated with each requester process. Therefore, Bayeh does not teach features specific to claim 2.

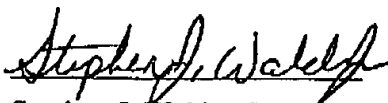
Therefore, in view of the above, Applicant respectfully submits that neither Cohen nor Bayeh, either alone or in combination, teaches or suggests each and every feature recited in dependent claim 2. By virtual of its dependency on claim 2, neither Cohen nor Bayer, either alone or in combination, teaches or suggests each and every feature in dependent claim 3. Thus, Applicant respectfully requests withdrawal of the rejection of claims 2 and 3 under 35 U.S.C. § 103(a).

III. Conclusion

It is respectfully urged that the subject application is patentable over Cohen et al. (U.S. Patent No. 6,434,618 B1) in view of Bayeh (U.S. Patent No. 6,223,202 B1) and is now in condition for allowance. The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

Respectfully submitted,

DATE: December 23, 2003



Stephen J. Walder, Jr.
Reg. No. 41,534
Carstens, Yee & Cahoon, LLP
P.O. Box 802334
Dallas, TX 75380
(972) 367-2001
Attorney for Applicant

SJW/im

GB9-2000-0096-US1

Spender
Enabling Multiple Client Access to a Process-Based
System or Program from a Single Java Virtual Machine

REPLACEMENT
SHEET !

1/2

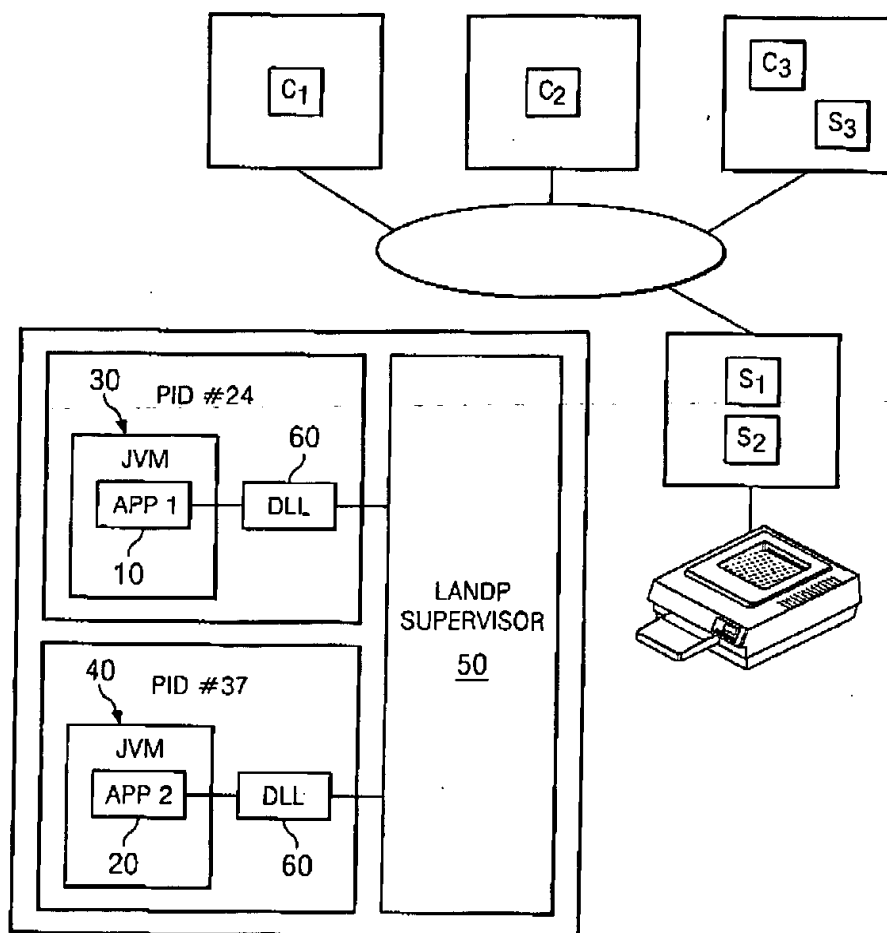


FIG. 1
(PRIOR ART)

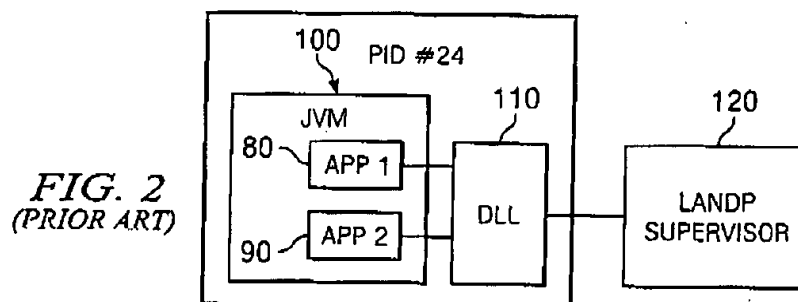


FIG. 2
(PRIOR ART)

GB9-2000-0096-US1

Spender

Enabling Multiple Client Access to a Process-Based
System or Program from a Single Java Virtual MachineREPLACEMENT
SHEET

2/2

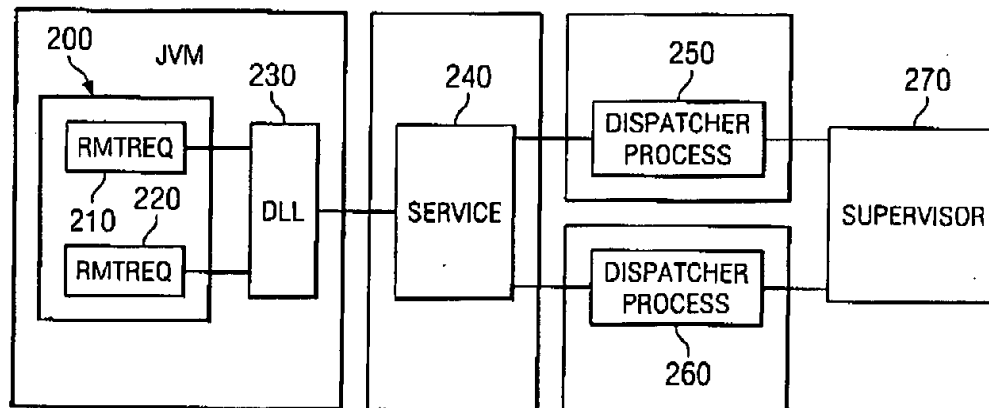
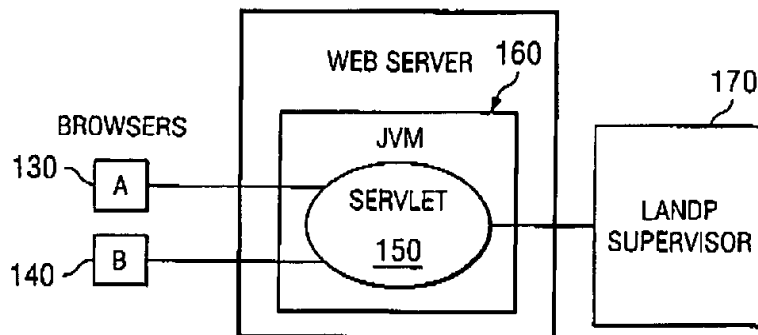


FIG. 4

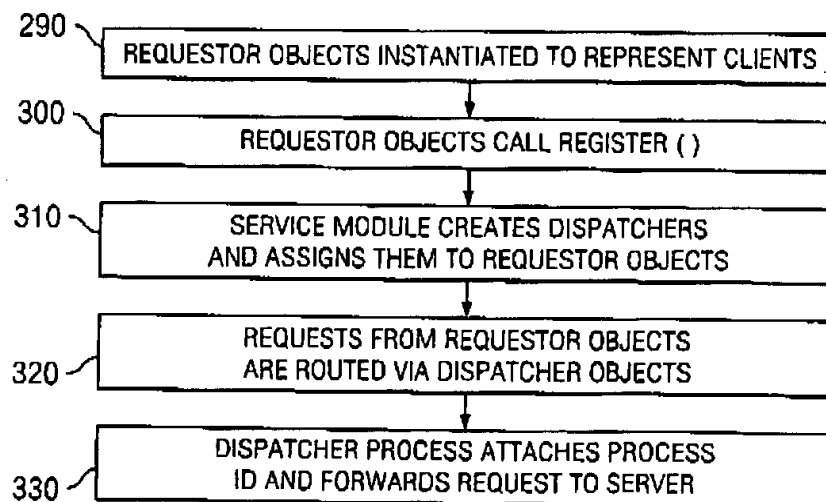


FIG. 5